APPENDIX A

Appendix A provides certain theoretic details of the modular constraint solver. The modular constraint solver is capable of finding all solutions to a given set of linear constraints under modular number system and expressing them in a closed form.

Definition 3 (Multiplicative Inverse of Bit-Vector): The *multiplicative inverse x* of an *n*-bit bit-vector *a* is defined as $\{x \mid (a * x) \text{ modulo } 2^n = 1\}$, and denoted as *multiplicative_inverse(a)*.

Note that while multiplicative inverse exists for every nonzero real number, in integral number domain only integers 1 and -1 have multiplicative inverses. In modulo-2^n number system, only odd numbers have one and only one multiplicative inverse. For example, for 3-bit-wide bit-vectors, 3 is 3's multiplicative inverse because 3 * 3 = 9 and 9 modulo $2^3 = 1$. On the other hand, 2 does not have any multiplicative inverses. Although no multiplicative inverse exists for even bit-vectors, the concept of multiplicative inverse can be extended to the *multiplicative inverse with product k.*

Definition 4 (Multiplicative Inverse of Bit-Vector With Product k): The multiplicative *inverse x* of an *n*-bit bit-vector *a* with multiplication product k is defined as $\{x \mid (a * x) \text{ modulo } 2^n = k\}$, and denoted as $multiplicative\_inverse_k(a)$.

For example, for 3-bit-wide bit-vectors, 3 is 6's multiplicative inverse with product 2 because 6 * 3 = 18 and 18 modulo $2^3 = 2$. Moreover, 0 does not have any multiplicative inverse with nonzero product, but every bit-vector is a multiplicative inverse of 0 with product 0.

As for the linear constraint example above, modulating the equation with 2^3 produces:

$$\begin{bmatrix} 1 & 1 \\ 0 & 5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

Solving the second row by $multiplicative\_inverse_2(5)$ results in $y = 2$. Substituting it back to the first row derives $x = 3$.

A bit-vector may have none or several multiplicative inverses. The following theorem gives the number of multiplicative inverses for a given bit-vector.

Theorem 1: Given a nonzero n-bit-wide bit-vector a with greatest odd factor a', it can be expressed as $a = a' * 2^m$, where m is an integer.

(T1.1) a has *exactly one* multiplicative inverse with product k if and only if a is an odd number; that is, $m = 0$. Moreover, $multiplicative\_inverse_k(a) = multiplicative\_inverse(a) * k$.

(T1.2) a has no *multiplicative inverse* with product k if and only if a is an even number and k is not a multiple of 2^m.

(T1.3) a has *exactly* 2^m *multiplicative inverses* with product k if and only if a is an even number and k is a multiple of 2^m.

For example, for 3-bit-wide bit-vectors, 6 (= $3 * 2^1$) has no multiplicative inverse with product 3 because 3 is not a multiple of 2^1, but has exactly two multiplicative inverses of product 4 as {2, 6}. Furthermore, all the 2^m multiplicative inverse bit-vectors in (T1.3) can be represented in a closed form as shown in the following theorem.

Theorem 2: Given a nonzero even n-bit-wide bit-vector a with the greatest odd factor a', it can be expressed as $a = a' * 2^m$. If k is a multiple of 2^m as $k = k' * 2^m$ and b is the only multiplicative inverse of a' with product k', that is, ($a' * b$) modulo $2^n = k'$, then the multiplicative inverse of a with product k can be represented in the following closed form:

$$multiplicative\_inverse_k(a) = (b + 2^{(n-m)} \bullet t) \text{ modulo } 2^n$$

where t is a *free integer* between 0 and $2^m - 1$.

For example, for 4-bit-wide bit-vectors, let $a = 6 = 3 * 2^1$ and $k = 10 = 5 * 2^1$, which is a multiple of 2. By (T1.3), it can be determined that $a = 6$ has exactly two multiplicative inverses with product $k = 10$. Because the multiplicative inverse of 3 with product 5 is 7 ($3 * 7 = 21$ modulo $2^4 = 5$), all the multiplicative inverses of 6 with product 10 can be presented as "$7 + 2^3 * t$", for $t = 0$ or 1.

The concept of multiplicative inverse is extended and applied to solve the linear bit-vector matrix constraints using the *Gauss-Jordan elimination method.* For the linear equation A * x = b, the solutions can be represented in a closed form as

$$x = N \bullet f + x_0$$

where

N null matrix (because multiplying it with the constraint matrix A will result in a zero matrix);

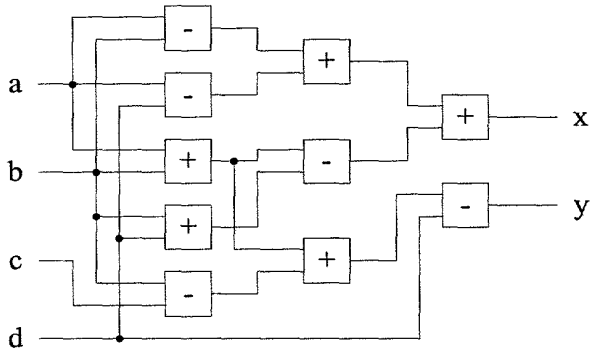
f column matrix containing some free variables;

x_0 solution and can be derived from A, N and b in linear time.

Applying different values of the free variables in f, different values of x is obtained, each of which is a solution of A * x = b.

EXAMPLE

The detailed procedure of the linear constraint-solving algorithm is illustrated using the following linear circuit example:



Assume all the signal buses of the above linear circuit example are 4-bits wide and the initial assignments for output $x = 2$ and $y = 10$ are given. The linear constraints can be expressed as an integer matrix equation:

$$\begin{bmatrix} 3 & -1 & 0 & -2 \\ 1 & 2 & -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 10 \end{bmatrix}$$

Modulating the coefficients by 16 (2^4) produces:

$$\begin{bmatrix} 3 & 15 & 0 & 14 \\ 1 & 2 & 14 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 10 \end{bmatrix}$$

First, a (series of) row operation(s) is performed to *upper-trianglize* the matrix; that is, make its lower-left element(s) 0. Here, the first row of the above example is multiplied by the multiplicative inverse of its first element (i.e., *multiplicative_inverse*(3) = 11) and subtracted from the second row. The equation then becomes:

$$\begin{bmatrix} 3 & 15 & 0 & 14 \\ 0 & 13 & 14 & 6 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

The partial solution for the last nonzero row is then solved. To obtain a particular solution for a single-row equation, a check to determine if the greatest common divisor (*gcd*) of this row has a multiplicative inverse with product of the constant vector on the right-hand side is performed. If the multiplicative inverse does not exist, there will be no solution for this row and, therefore, no solution for the entire matrix equation. For the above example, the *gcd* for the last nonzero row {13, 14, 6} is equal to one, which has a multiplicative inverse with product 4. All but one variable (*b* and *c*) is set to zeros and the value of *d* equal to $multiplirative\_inverse_4(6) = 6$ is derived. Therefore, a particular solution for the last row is:

$$\begin{bmatrix} b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

The next step is to obtain the null matrix for this single nonzero row matrix. For a row matrix with *k* nonzero elements, solving its null matrix is equivalent to finding (*k* - 1) linearly independent column matrices and each of the column matrices can produce a zero matrix by multiplying it with the row matrix. These (*k* - 1) linearly independent column matrices can be generated by selecting (*k* - 1) pairs of nonzero elements in the row matrix and for each pair, the multiplicative inverse of one element with product as negation of the other is solved. For the last row in the above example, two pairs of nonzero bit-vectors: (13, 14) and (14, 6) are selected. Finding the multiplicative inverse generates $multiplicative\_inverse_{13}(-14) = 10$ and $multiplicative\_inverse_{14}(-6) = 3$. The null matrix is obtained by assigning each column with an element equal to the multiplicative inverse and the other element to one.

For the above example, the null matrix for the last row is:

$$N_1 = \begin{bmatrix} 10 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix}$$

Therefore, a partial solution set for variables (b, c, d) is:

$$\begin{bmatrix} b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

where i and j are free variables between 0 and 15.

Once the partial solution set for the last row is generated, the corresponding variables in the next nonzero row can be substituted with the solution and the solution set cab be refined to satisfy the last two nonzero rows of A. This process is iterated until the first row is reached. This results in the following closed form solution for the entire linear constraints:

$$3a + [15 \;\; 0 \;\; 14] \cdot \left(\begin{bmatrix} 10 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix} \right) = 2$$

$$\Rightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 14 & 6 \\ 10 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 10 \\ 0 \\ 0 \\ 6 \end{bmatrix}$$

APPENDIX B

Appendix B provides an exemplary procedure that can be utilized by the modular constraint solver to solve linear equations having bit slicing and concatenation operations. The procedure involves enumeration of some of the variables and applies the constraint implication to screen out the illegal set of solutions.

As an example, consider the following linear constraints in the matrix format:

$$\begin{bmatrix} 4 & 4 & 5 & 4 \\ 3 & 9 & 14 & 5 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 9 \\ 11 \end{bmatrix}$$

Furthermore, the input variables a, b, c, and d are 1-, 2-, 3-, and 4-bit wide, respectively, and are the bit-sliced signals of some variables. Moving the variables a and b to the right-hand side creates the following:

$$\begin{bmatrix} 5 & 4 \\ 14 & 5 \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} 12 & 12 & 9 \\ 13 & 7 & 11 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

Multiplying both sides of the above equation with the *inverse matrix* of

$$\begin{bmatrix} 5 & 4 \\ 14 & 5 \end{bmatrix}$$

to obtain a constraint of c and d in terms of inputs a and b as:

$$\begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 12 & 5 \end{bmatrix} \cdot \begin{bmatrix} 12 & 12 & 9 \\ 13 & 7 & 11 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & 2 & 3 \\ 1 & 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

After the above transformation, the modular constraint solver can express the variables with larger domains (e.g., c and d with 3- and 4-bit wide) in terms of those with smaller domains (e.g., a and b with 1- and 2-bit wide). In the decision making process, the modular constraint solver can then enumerate the variable with a smaller number of

choices followed by implications using the above equation to check if the nonlinear constraints are satisfiable. In the above example, the modular constraint solver can determine that if $a = 1$, then $c = 2 * b + 9$, which will always be greater than 8 and, thus, contradicts its domain constraint (i.e., c is 3-bit wide and, thus, $0 \leq c \leq 7$). Therefore, it is necessary to have $a = 0$, which results in the following four solutions: $\{a,b,c,d \mid (0, 0, 1, 9), (0, 1, 1, 4), (0, 2, 1, 15), (0, 3, 1, 10)\}$.